IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	10/717,698
Applicants	:	Joseph A Pruitt *et al.*
Filed	:	November 20, 2003
Title	:	Method And System For Using Feedback in Accessing Network Services
TC/A.U.	:	3628
Examiner	:	D. Vedder
Docket No.	:	812495-000220 (10.83)

DECLARATION OF JOSEPH A. PRUITT UNDER 37 C.F.R. § 1.131

Dear Sir:

I, Joseph A. Pruitt, a named co-inventor of pending U.S. Patent Application No. 10/717,698, entitled "Method And System For Using Feedback in Accessing Network Service" hereby declare:

1. I am a co-inventor, along with Gary Mager, named on the United States Patent Application No. 10/717,698 ("the '698 application").

2. The '698 application generally pertains to a method for allowing feedback from previously accessed services for responses to later requests for services. We invented the subject matter of the claims of the '698 application prior to the priority date of the cited U.S. Publication No. 2004/0122926 ("Moore") while we worked at F5 Networks, Inc ("F5").

3. As detailed below, the subject matter of the '698 application was conceived prior to December 23, 2002, the priority date claimed by Moore.

4. Exhibit A is a true copy of pages of e-mail exchanges between myself and Mr. Mager regarding the subject matter of the '698 application that were prepared and exchanged in accordance with normal business practices while at F5. Exhibit A, p. 1 shows descriptions of the embodiments of the claimed invention in an e-mail written by Mr. Mager. For example, Mr.

Mager first suggested the concept of collecting feedback data such as quality of service and reliability from a service provider. The e-mail includes comments from myself conceiving the concept of providing feedback data and basing the selection of the next service request based on the feedback. (Ex. A, p. 1). Follow up e-mails from Mr. Mager suggest the process of secondary service providers to provide the web services in pp. 2-3 of Exhibit A. The final e-mails in the chain are from Mr. Mager and include the concept of collecting feedback data for future requests of the service. (Ex. A, pp. 5-6).

5. In view of Exhibit A, the subject matter of the '698 application was conceived prior to December 23, 2002, the priority date of Moore. Additionally, from prior to December 23, 2002 to at least until November 20, 2003, the filing date of the '698 application, we also continued to diligently investigate and refine the technology that is the subject matter of the '698 application.

6. This included work performed by myself from the conception date until November 20, 2003. After the conception of the subject matter of the '698 application, I investigated possible platforms that could operate the subject matter of the '698 application including our F5 Big IP product. After my initial investigations I concluded that I should focus on the development on extending open source UDDI servers with the subject matter or incorporating the subject matter with Microsoft's UDDI server that was included in Windows Server 2003. In this process, I reviewed product literature and open source code (if available) for those products to attempt to determine the likely proper UDDI server. Ultimately, I selected an open source UDDI server as a platform for development of the subject matter of the '698 application.

7. I then designed the data collector and data repository components that determined the metrics and reliability of information sources requested. The prototyping work I performed included developing the C++ based "Data Collector" component to store/retrieve metric/feedback data from within a "Data Repository" which was a text file. I initiated this process by reviewing the open source code for extensibility points. I then wrote a daemon with a network interface for the Data Collector component in C++.

8. Once I developed the prototype code in the form of the Data Collector and Data Repository components, I proceeded to build a unit test framework to emulate the connections from the UDDI server and directly from a client. Once I finished these components, I tested the Data Collector and Data Repository components to determine whether the subject matter could be performed in practice. This testing included simulating data for metrics such as response data and latency data.

9. After testing and refining the components, I determined to my satisfaction that the subject matter could be operated in a practical manner. I then began to assist our patent counsel in drafting a patent application. My work on incorporating the subject matter in prototype code and assisting in drafting the patent application continued through at least November 20, 2003.

10. After the filing of the present application, my work was abandoned on the prototype code because F5 decided not to incorporate the subject matter in a product. I have attempted to locate additional documents and electronic materials relating to my work on the prototype components. It is my personal practice to recycle both documents and hardware for older projects and therefore I have been unsuccessful in trying to find any additional documentation.

11. In addition to our work on the technology, from prior to December 23, 2002 to November 20, 2003, we continued to diligently work with Patent Counsel on developing the patent application that resulted in the '698 application. This included the e-mails in Exhibit A which evidence conception. This is also shown by the e-mail correspondence attached as Exhibit B which reflects the drafting of a written disclosure that we sent to our outside Patent Counsel. Exhibit C are e-mails that reflect correspondence that evidence the drafting of the application up until November 20, 2003.

12. Our Patent Counsel proceeded to file the application on November 20, 2003.

13. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and, further, that these statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the '698 application or any patent issued thereon.

Dated: June 15, 2009

Joseph A. Pruitt

EXHIBIT A

REDACTED

-----Original Message-----
From: Joe Pruitt

To: Gary Mager
Subject: RE: Article on the use of SOAP-based intermediaries

comments below...

-Joe
-----Original Message-----
From: Gary Mager
Sent:
To: Joe Pruitt
Subject: RE: Article on the use of SOAP-based intermediaries

Joe,
I was reading a bit about Web services, and have a question. Is the concept of costs of services, and making services decisions based on costs, a topic of discussion (or is it included within UDDI?). Assuming that at least some services charge for their info/services, a service user would want to find a web service provider with the lowest cost. There are many models for pricing - including a published price list, dynamically negotiated pricing, automatic auctions, bundling, etc. An intermediate network device could automatically find the best price. In fact, one could provide a web service that finds the best price on services, buys services in volume and passes the discount on (a Costco of Web Services?).

<Joe>
It definitlely could be a topic of discussion. UDDI is really only a registry (what services are available and where to get them). It doesn't address other decision making questions (QoS, Cost, etc). There are websites that do this for consumer products (http://www.mysimon.com, etc), but nothing as of yet for web services. Really, this would be developing a new application but the interface would be webservices. Very interesting. Probably not much we could do from a routing perspective, but if we could claim some IP on this, it could be useful in the future when consumer based webservices take off. QoS falls in here also. Cost of services is a factor, but you also don't want to buy from "Joe's Garage Outlet" just to save a couple of cents. Reliability, Rating, and QoS are points to look at.
</Joe>

There's also the concept of link cost load balancing for web services - find the lowest dollar cost links to transfer info. Could the Internet some day evolve so that not only ISPs, but intermediate links bill for their use, and intelligent routing needs to consider the dollar costs of different links along the way? I confess to a little ignorance here. I know that once upon a time, the infrastructure of the Internet was free. Are there charges now for Internet infrastructure, beyond the ISP?

<Joe>
Now, this is more in our possible product space. I don't currently know of anything out there that has a charge by the link (outside of normal ISP charges) with regards to layer 7 traffic. With regards to webservices, most applications now are being developed on a single LAN. Once adoption increases for webservices and method invocations are starting to make their way out of a single local lan and out across the internet, this can be very appealing. Not only for a cost saving perspective, but a QoS one as well. If you have 2 links, one a 56k, the other a T1 and want to send a message across, the decision needs to be made as to the balance of cost and QoS. Maybe our Link Controller could play in here.
</Joe>

Oops, that was more than a question.
-gary

REDACTED

-----Original Message-----
From: Gary Maaer
Sent:
To: Joe Pruitt
Subject: RE: web services broker

Though we want to focus the patent process on our products, we shouldn't limit them there. Many strategic reasons for this, but I'll save that for another discussion.

I wasn't thinking of consumer services, but services for any network node that needs info/services, and the commercial ones are the most interesting. Unlike the consumer sites, in webservices, you don't want to merely gather info, sort it, and present it to a user. You want the intelligence to make decisions based on constraints and preferences, in the same way that BIG-IP and 3DNS do. So you have these network devices that combine 3DNS and BIG-IP intelligence, but are publicly accessible. I'll call the device a "broker" (Costco was a bad analogy).

1. A process somewhere has a need for a service (or information).
2. It sends a service request to the broker, either together with the desired constraints/preferences or the c/p is already known from a prior communication.
3. The broker accesses lots of data to make a routing decision. Maybe it queries one or more registries. Maybe it queries some target services. Maybe the querying was done ahead of time and the broker has the data ready.
4. The broker decides the best service provider, based on the preferences - looking at QoS, cost, reliability, rating, etc., also link costs. Maybe in the same way that certificate authorities vouch for a certificate, a rating authority rates service providers. An example of a constraint/preference would be to select the lowest cost service provider with a rating of at least 80.
5. The broker forwards the service request to the best provider.
6. The broker collects a service charge. Or maybe the broker is available by subscription.

Now suppose the Internet has a lot of traffic, and data transfer is sluggish or unreliable. The service requester needs a lot of data transferred quickly, has specified a preference that data transfer time is the highest factor, provided it doesn't increase costs by N%. The broker sees that there is a commercially controlled toll pipe available, for a reasonable charge. So now it makes 2 decisions: one is to use the toll pipe; two is to use a service that is close to this pipe, in order to further reduce transfer time.

There are a million scenarios, but how about some basic claims like:

1. In a network device on a WAN having a requestor and a plurality of service providers at remote locations, a method comprising:

- receiving, from a remote requestor, a request for a service;
- receiving, from the requestor, one or more service preferences;
- receiving data describing attributes of the remote service providers, the attributes including cost, reliability, QoS ...
- determining an optimal service provider;
- forwarding the request to the optimal service provider.

2. The method of Claim 1, wherein determining an optimal service provider comprises:

- determining a cost of a link to each service provider; and
- determining an optimal service provider based on the cost of the links to each service provider.

REDACTED

-----Original Message-----
From: Gary Mager
Sent:
To: Patrick Jenny; Rick Masters; Joe Pruitt; Dave Schmitt; Bryan Skene
Cc: *Invention Archive
Subject: Secondary providers of Web services

Following up on our patent discussion of a traffic manager that brokers web services by matching requests with service providers, based on a set of user preferences, I'm wondering if we can patent an idea as follows:

A method of managing requests, comprising:
receiving a request from a client;
forwarding the request to a provider;
determining whether the provider provided a desired response;
if not [error, timeout, etc.], forwarding the request to a second provider.

Additional elements are handling the accounting, determining what types of requests are eligible for forwarding, authorization, etc.

I don't know if BIG-IP does this or not. Best I can tell from the docs is that BIG-IP handles errors by not directing FUTURE requests to the problem server, but I can't tell whether it resends the failed request to another server. In any case, doing this for web services seems more important, so that a single error doesn't stop a complex chain of processing.

This opens up an area of secondary providers of Web services - providers who are willing to provide services when a primary provider has a problem - kind of like secondary providers of components for manufacturing. Service providers should like this, since it saves their ass when they have a problem - like being overloaded, and they can get by with less capacity or redundancy. Clients, of course, not only like this because there are less errors, but because they don't need to pay a premium for high reliability if they know their request will get forwarded to a secondary provider.

Of course, this doesn't solve the problem if the request is part of a stateful communication. There are ways to address that that could be added.

Is this novel? Is it useful?
-gary

REDACTED

-----Original Message-----
From: [redacted]
Sent:
To: Gary Mager; Patrick Jenny; Rick Masters; Dave Schmitt; Bryan Skene
Cc: *Invention Archive
Subject: RE: Secondary providers of Web services

This is interesting as I currently don't know of any TM devices out there in the WS/XML space that specifically broker and provide this kind of reliable messaging. Basically, this would extend our current load balancing to allow for retries to alternate backend servers depending on the characteristics of the current server response (or lack of) without having to rely on external health checkers. I think that's pretty novel as well as useful. Right now BIG-IP uses several means to load balance client requests to servers and after it's decision is made as to where to route the traffic the client will be directed there and that's the end of it. This functionality would add the extra steps of validating the results and then deciding as to whether to route the message back to another server. With regards to the full proxy architecture, a plugin could be created to inspect the outbound traffic and could flow the packet back to the inbound path with a specified target server (or alternate LB method). Pretty cool.

-Joe

-----Original Message-----
From: Gary Maaer
Sent:
To: ...litt; Patrick Jenny; Rick Masters; Dave Schmitt; Bryan Skene
Cc: *Invention Archive
Subject: RE: Secondary providers of Web services

REDACTED

With Joe's endorsement, I am submitting this as a potential patent idea, to be voted on or off the island at the next patent committee survivor meeting.

With respect to retries and persistence problems when a server is maintaining state info, here are some thoughts. Some of what follows is not limited to retries. Even though they involve server-side changes, if they are novel, they would be good to patent, to have coverage on the server. Also, recognizing that you all have much more expertise on this subject than I do, to the extent that these are dumb ideas, my apologies for wasting your time.

1. A server (and traffic manager?) should distinguish requests that require persistence from those that don't. Obviously, first requests don't. Also, within a persistent session, there must be numerous requests that don't require persistence, even if a server is maintaining state info. So if a BIG-IP can distinguish the latter requests, it need not persist on them. (The question of how is TBD, and requires some additional intelligence.)

2. A BIG-IP and a server recognize the requests that don't require persistence and treat them as a lower priority. e.g. If the server can't handle it, returning an error isn't so bad if BIG-IP will retry with another server. The error is more like "pass this on to someone else." So a server having a surge of requests might reject these lower priority ones, but handle the stateful requests. This architecture certainly makes sense for sites that download a lot of data. There should be high priority servers that do the stateful transactions, then non-stateful ones that merely download the data. Without knowing, I would hope that web sites already do this, but maybe not.

3. How a BIG-IP can recognize requests that don't require persistence is an interesting problem, with possible patentable solutions. A server can send cookie data that tells when a stateful transaction is starting, and BIG-IP can recognize this to mean the beginning of persistence, but that doesn't address the non-stateful requests such as in #2 above. More work would be to have a server including data in URLs that indicate whether state is needed or not, and BIG-IP can recognize this. A patent claiming load balancing a request even when persistence information is available would be interesting. Currently, if the persistence data (e.g. cookie) is there, BIG-IP always persists (I think), which is overkill.

4. To the extent that this requires work on the server side, F5 can:
-patent it
-Provide server plug-ins or libraries to make it easier. Sell or give these away to F5 customers.
-I suppose there's an interesting AI problem that involves observing program flow of a web server, recognizing the URLs/requests that require persistence, and marking them as such, but this is beyond my thought processes.

Now, one nice thing about patenting a first solution to a problem is that a patent could potentially cover all future-developed solutions. Imagine, for example, if someone patented the first solution of persistence and had a claim that covered LB & persistence based on data received with a request. This would cover all persistence. So a claim that covers selective persistence even when persistence info is available could cover many solutions developed in the future. Similarly with a claim covering a TM that does retries, or other inventions. Even if our proposed solution isn't the best one, a broad patent could cover the better ones that come along later. This, of course, is another plug for getting those Web Services ideas flowing now.

-gary

REDACTED

From: Gary Mager
Sent: .
To: Patrick Jenny; Joe Pruitt; Rick Masters; Dave Schmitt; Bryan Skene
Cc: *Invention Archive
Subject: RE: Secondary providers of Web services

Another issue is the granularity of requests. If someone is streaming a movie, and the streaming server has a problem, a TM should be able to transfer the streaming to another server/provider without missing a beat, by passing the appropriate parameters. This can apply to any file download, where the file can be segmented.

A TM can not only monitor for errors, but receive dynamic "complaints" from clients, when there are problems. The complaints can not only be used in future evaluations, but change an existing transaction. So if a client is getting interruptions when receiving a streaming video, the client automatically sends a complaint to the TM, which can then transfer to another server/provider.

-g

EXHIBIT B

REDACTED

-----Original Message-----
From: Gary Mager
Sent:
To: Joe Pruitt
Subject: RE: 10.83 - Feedback Mechanisms with Network Services

Joe,
I think this is good for a disclosure. I inserted some comments, which are just quick thoughts -- feel free to reject any/all of them. I got a little tired& hungry by the last scenarios, so I didn't read them closely.

Diagrams will be useful, but I think we can start with the ones from the previous application, and in the meeting talk about any variations.

There are a lot of variations and lists here, which is good to get a broad patent, but it adds complexity, and could make it hard to focus. I like the way you organized it into types of feedback, where it is stored, how it is used, etc. I think we'll want to expand the process section into one or more flow charts that show possible points where feedback is collected, transmitted (or shared), and used.

More thoughts:

In one scenario, directory services are themselves services (the hierarchical system from the previous application). A client requests from the root resolver a lookup for service X. The root resolver performs the lookup for secondary resolvers that know how to look up service X. Using various feedback mechanisms, it finds that resolver Y has performed the best when handling requests for service X. It may then automatically forward the client request on to resolver Y, and either by observing or by later feedback, adds feedback info on the new transaction with resolver Y.

Similarly, in one scenario, a retail service provider receives requests for service X from various clients. The retail provider uses client criteria + feedback to locate wholesale providers, and acts as an intermediary in each transaction. An advantage of the retailer is that if the wholesale provider has a temporary problem that isn't discovered until the transaction begins, the retailer can detect this and switch to the next provider, in a way that's transparent to the client. The retailer competes for clients on response time, reliability, etc. in the same way as the simple example. It allows the clients to make decisions based on the reputation of the retailer, which might use unknown wholesalers.

Let's discuss P2P with John, but not add it to your document. I'm torn between not wanting to get off the track of the invention, and thinking it has some cool possibilities.

-gary

-----Original Message-----
From: Joe Pruitt
Sent:
To: Gary Mager
Cc: Joe Pruitt
Subject: 10.83 - Feedback Mechanisms with Network Services

Gary, I'm going to flush this out tomorrow, but if you could provide any feedback before the end of the day, that would be great. I didn't include the P2P hooks. Let me see if you can fit that in anywhere.

Let me know if this is an OK format for a first disclosure. Also, if adding diagrams would be useful or not at this point. I tried, but they were either very simple or way to complicated.

-Joe

EXHIBIT C

-----Original Message-----
From: Gary Mager [mailto:g.mager@F5.com]
Sent:
To: Joe Pruitt; jjardine@alumni.washington.edu
Cc: John Jardine
Subject: RE: 1000.4US01/10.83

REDACTED

See comments in whatever color this is below.

-----Original Message-----
From: Joe Pruitt
Sent: Monday, July 28, 2003 2:01 PM
To: Gary Mager; 'jjardine@alumni.washington.edu'
Cc: 'John Jardine'
Subject: RE: 1000.4US01/10.83

I agree with Gary in that we should go the abstract route, otherwise things are going to get very complicated very quick. I've reviewed John's initial stab and I don't see anything that stands out from what Gary has already commented on. I don't want to hold you up any longer on the initial draft. A few comments are included below.

-Joe

From: Gary Mager
Sent:
To: 'jjardine@alumni.washington.edu'; Joe Pruitt
Cc: John Jardine
Subject: RE: 1000.4US01/10.83

J & J,
Here are some thoughts.

-Seems like a lot of configurations. I guess this is what we discussed, though. I wouldn't spend too much time discussing all of the variations.

-In attached figure 1, I made a generalized configuration. The data collector & repository are modules, and can either or both can reside in the client or the directory service (or in a proxy). The collector can also be in the service provider. If we abstract out these modules from their physical location, it might be easier to discuss them in a generalized way. Just a thought.

<Joe>
Like I said above, I like this approach a lot. One thing I might add is that the SP could contribute to the data collector and repository as well. With that being said, Isn't the DS and SP essentially the same thing. The DS is kind of SP that returns other service locations prioritized by feedback. The SP prioritizes service processing based on the feedback as well. I don't know if going this route get's too generic and encroaches on the "browser-webserver"'s use of HTTP cookies. Creating this like a 2 way conversionaion simplifies this quite a bit and we can list the interaction client-DS, client-SP as a overlapped implementation. Then we can include a extension of allowing the Client to communicate with other clients to build a client side "knowledge base" and the SP can talk to another SP to pass around the knowledge it gains.

Is this too generic? Or, do you all think it's best to continue on the three way idea (DS + Client + SP) in one system?

</Joe>

[gm] I think this shows the difficulty of focusing on physical configurations, rather than functional modules. Potentially, any physical component can take on different functional roles, do data collection, processing, sharing, etc. But I'm concerned a written description and the claims will be confusing if we say that every component is essentially the same thing. Generally, in our applications that show clients and servers, we say that these are functional roles, and that any client can also act as a server, etc. Then the description is a little more concrete, while preserving the abstract. The description of 2-way conversations does simplify it, but I'm concerned that if we describe it as merely 2-way conversations, we rub too close to prior art, which includes 2-way conversations, where each server sends a cookie to the client with client-server specific data for its own later use. But in 3 way, the DS sends cookie-like data to the client with data pertaining to the client's interaction with a third party (SP), or the SP sends data to the client for the SP's later use, or data from multiple devices are stored somewhere for the SP's later use. So, similar to the way cookie persistence used a 3-part system to expand on the use of cookies, this invention expands the use of cookies (or generalized data blobs) to another 3-part system.

BTW, your point about the SP prioritizing service processing based on feedback is a good one. I don't recall hearing that before.

- The 3rd box in Fig. 7 says the DS returns the results + additional feedback data. I thought that the DS wouldn't have the additional feedback at this point, so it couldn't be combined. If the DS sends feedback data, I think it would happen separately, after the client access the service.

<Joe>
The DS can include data here that it wants passed on to the service provider through the client. (ie. Location of DS, client id token, etc). This data isn't really "feedback" but other "stuff" that it wants to float around in the virtual circuit we are creating. I think making this an "additional" option at this point is valid. Do we want to limit the DS from returning data on the initial conversation?

</Joe>

[gm] I agree, if we expand feedback to include other stuff. I was thinking narrowly. Maybe we need 2 terms. Feedback could be specific to feedback about a transaction in a narrow sense, and XYZ could be feedback+other stuff. The reason I say this is that the use of feedback in ways we are talking about adds a lot of novelty, where a server sending general data (e.g. client id) to a client to be returned later gets close to prior art. In our claims, I'd be happy if we can get claims covering various uses of feedback, in the narrower sense.

- Fig. 7 seems to have too much implementation-specific detail, for a high-level figure. An ordered list is just one implementation, and not important to this application. It would be more general to just say "results", or some other general term, and leave the ordered list for discussion of one implementation (copied from the previous one). Attached fig 2 is a generalized flow chart, that might be used in place of, or in addition to, fig. 7. It starts where fig. 5 from the 10.72 application leaves off, after the service location(s) are returned to the client. One of my thoughts is that, though fig. 7 is accurate, it might simplify it to unroll the loop and have the first request without feedback, and the second

with feedback (since one reading fig. 7 block 1 might wonder where the original feedback came from).

Also, in my fig. 2, I tried to reference the data collector and repository generically, without saying where they reside, so the figure covers all configurations. My fig. 3 then shows a flow where collectors reside in 3 different places. This can be used to say that a collector can reside in any one or more of the 3 (actually, a fourth place would be a proxy or traffic manager).

If my fig 2 was used as a general flow, then fig. 7 could be a specific configuration/implementation. In any case, I like the idea of having a general flow chart that applies to all of the claims, with the claims moving "in" from the figure, rather than "out" from the figure.

Note that the colored blocks in the fig are not meant to actually be there. The yellow shows an implementation when the client collects and stores feedback data; the green shows where the directory service collects and stores. Either of these could be in a fig that is a variation of fig. 7.

- Claim 8: Sounds a bit like BIG-IP or 3-DNS. Now if it said that the Client accumulates the network information, that would help distinguish it. Similar issue with some of the other claims. I think there's a significant step when we distinguish general information, such as network traffic, health, etc., from info specific to a particular transaction. A second distinguishing factor is having the Client automatically collecting the data. A third is a difference in the type of data -- distinguishing service-specific data to general (network, response time, etc.).

- My fig. 4 shows the aspect of the invention where the service provider receives feedback on request results, and automatically modifies its attributes based on the results (to be more competitive, or to charge more, if it is underbidding). It would be good to have some claims covering these steps.

- Fig. 5 is a variation from fig. 4, where the S.P. receives feedback data pertaining to its own transaction, rather than the lookup results, and automatically modifies its attributes based on that. A special case of this is when the client feedback is part of the transaction.

<< File: 10.83 - more figures.zip >>

-gary

-----Original Message-----
From: John Jardine [mailto:jjardine@alumni.washington.edu]
Sent:
To: Gary Mager; Joe Pruitt
Cc: John Jardine
Subject: 1000.4US01/10.83

Gary and Joe:

Attached you will find a first draft of claims and figures for the above-identified application. The figures have text in them to indicate what aspect of the

invention they are directed to. This text will be removed by the final draft of the application. I am ready to begin drafting the application as soon as I receive word from you.

John
John S. Jardine
116 129th Ave. NE
Bellevue, WA 98005
Telephone: (425) 467-5686
Fax: (425) 462-6746
hightechlaw@mindspring.com <<mailto:hightechlaw@mindspring.com>>